Flow-chart

Introduzione agli algoritmi e ai diagrammi a blocchi

ugo.rinaldi@gmail.com

Sommario

- Fasi dello sviluppo di un programma
- Algoritmo
- Dati: Variabili e Costanti
- Le operazioni tipiche
- Diagramma a blocchi
- Simboli
- Le strutture condizionali
- Le strutture alternative
- Esempio

Il paradigma dell'informatica

Dal problema al programma

1. Definizione del problema
2. Analisi
3. Stesura dell'**algoritmo**
4. Minutazione in linguaggio di programmazione
5. Compilazione >>> programma oggetto
6. Se ci sono errori sintattici
   ritorna al punto 4
7. Linkaggio >>> programma eseguibile
8. Esecuzione
9. Test
10. Se ci sono errori logici
    ritorna al punto 3 o 4
11. Fine

Sviluppo del software

Flow-chart o diagramma a blocchi

Il diagramma a blocchi (flow-chart) è la

Rappresentazione grafica dell'algoritmo.

Algoritmo?

Algoritmo

Definizione :

Insieme di passi ordinato, finito, non ambiguo per risolvere una classe di problemi con i dati a disposizione

- Ordinato
- Finito
- Deterministico
- Generale

Esempio

Prepariamo il caffè:

1. Mescere l'acqua nella base
2. Inserire l'imbuto e mettere il caffè
3. Chiudere la caffettiera
4. Mettere la caffettiera sul fuoco lento
5. Appena uscito, spegnere il fuoco, gustare il caffè
6. Rimuovere e gettare la cialda

Perché ordinato?

Prepariamo il caffè:

1. Chiudere la caffettiera
2. Mescere l'acqua
3. Appena uscito, spegnere il fuoco, gustare il caffè
4. Mettere il caffè macinato
5. Mettere la caffettiera sul fuoco
6. Inserire l'imbuto

Riepilogo Preparazione del caffè

- DATI a disposizione:

Acqua, Fuoco, Caffè macinato, Caffettiera

PREPARAZIONE

1. Mescere l'acqua
2. Inserire l'imbuto
3. Mettere il caffè macinato
4. Chiudere la caffettiera
5. Mettere la caffettiera sul fuoco
6. Appena uscito, spegnere il fuoco

- RISULTATO: Caffè da gustare

I problemi e soluzioni (algoritmi)

Saremo in grado di risolvere problemi semplici o già affrontati.

Ogni soluzione ai problemi risolti andrà studiata, riletta ed assimilarla totalmente.

Saremo in grado di risolvere anche problemi complessi spesso scomponibili in problemi più piccoli e più semplici.

Le soluzioni sono realizzate manipolando dati ed operazioni di diversa natura sui dati

I Dati

- Oggetti utilizzati dalle istruzioni hanno un **tipo**
  - Numerico
    - Intero 0 ; 15 ; 7 ; 100 …
    - Reale 1,618 ; 2,7182818284 ; 3,141592 …
  - Alfanumerico F1 ; Under18 ; Rock86 ; Paola ; Catania
- **Variabili**

  contenitori di valori che possono variare;

  rappresentati da un **nome** simbolico;

  risiedono in memoria.

  Esempio: Nome, Colore, Importo, Somma, Numero
- **Costanti**

  dati invarianti rappresentati dal loro valore (possono anche essere rappresentati da un **nome** simbolico). Quelle alfanumeriche sono riportate tra apici o doppio apice. Per distinguerle dalle variabili. Le troviamo normalmente nelle espressioni.

  Esempio: 2, 100, "CATANIA", 18/09/2001, "VENERE"

Variabili e contenuti

	Numero	Somma	Altezza	Base
	150	**27000**	**15**	**1,5**
Città	**Acireale**			
Nome	**Eleonora**			
Professione	**avvocato**			

Classificazione dei DATI in base all'uso

- In ingresso

  i dati in input da conoscere per risolvere il problema

- Di lavoro/elaborazione

  valori intermedi calcolati prima di arrivare alla soluzione

- In uscita

  i dati comunicati all'esterno e la soluzione del problema

- Gli ultimi 2 spesso coincidono

Le variabili di Memoria

Le variabili possono essere scritte o lette.

- Quando vengono scritte:
  - -memorizzano un valore;
  - -perdono quello precedente (assegnazione)
  - -Una istruzione scrive in una sola variabile
- Quando vengono lette:
  - -Rendono disponibile il valore contenuto
  - -rimangono inalterate

Le operazioni standard

- Input
- Assegnamento
- Output
- Istruzione condizionale
- Istruzione alternativa
- Istruzione iterativa

Immissioni INPUT

- Consente di introdurre valori nella memoria del programma
- Memorizza un dato nella variabile specificata
- Normalmente si tratta di immissione da tastiera
- Distrugge il valore preesistente nella variabile

- Diverse istruzioni equivalenti:
  - Immetti A
  - Leggi BASE
  - Inserisci B

Esempi di INPUT

- Input a

A 150

- Input nome

nome MARTA

- Input colore

colore VERDE

Assegnamento

- Variabile ← costante|variabile|espressione
- Memorizza sempre un valore in una variabile

  Assegna 100 a IMPORTO:

  IMPORTO ← 100
- Il valore può essere il risultato di una espressione

  P ← 100 * 2 (a destra mai solo costanti)
- L'espressione può fare uso di costanti e/o variabili:

  X ← (Y / 100 * P) / (A – B / 5 * C)
- Le variabili a destra della freccia vengono solo lette
- Distrugge il valore preesistente
- Legge il contenuto di X, lo moltiplica per 2, lo assegna a X:

  X ← X * 2

Assegnazioni particolari

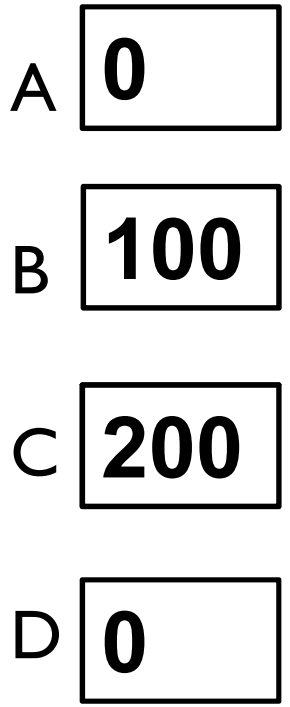
- Inizializzazione
  - Assegna un valore ad inizio programma, prima di un ciclo o prima di un blocco di codice
  - a=1
  - b=10
  - a=f
- Azzeramento
  - c=0
- Incremento
  - i=i+1
- Accumulazione
  - s= s + importo

Esempi di assegnazioni

- A ← 0
- B ← 100
- B ← A + C

attenzione alle variabili non inizializzate*

- C ← B * 2
- …
- D ← 0



*inizializzazione=impostazione iniziale di una variabile

Emissione OUTPUT

- Consente di far uscire valori dalla memoria del programma
- Rende visibile all'utente un valore letto da una variabile specificata
- Normalmente si tratta di emissione a video
- Non memorizza alcun valore
- Non distrugge il contenuto delle variabili

Esempi di OUTPUT

- Stampa "Immetti un numero"
- Visualizza B
- Stampa "La somma è ", B
- Scrivi 150
- Scrivi 'CIAO'
- Stampa C
- Stampa 'Quanti numeri vuoi generare?'

```
Immetti un numero
89
La somma è 89
150
CIAO
423659
Quanti numeri vuoi generare?
```

Rappresentazione di algoritmi

- Pseudo codifica
  - Linguaggio formato da parole del linguaggio comune
  - Può essere molto simile al linguaggio di programmazione
- Esempio:
  - Immettere da tastiera un numero in a
  - Calcolare il quoziente e memorizzarlo in q
  - Calcolare il resto e memorizzarlo in r
  - Stampare il resto della divisione
  - Azzerare c
  - Inizializzare x a 10

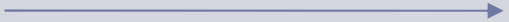
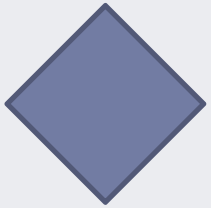
Il Diagramma a blocchi

Il diagramma a blocchi (flow-chart) è la

Rappresentazione grafica dell'algoritmo.

- Fa uso di simboli, collegati da linee con verso (frecce), che rappresentano il flusso logico dell'elaborazione.
- Ad ogni operazione corrisponde un simbolo diverso

I simboli

Inizio/Fine	**Inizia e finisce un diagramma**
	Input = immissione di dati dall'esterno (generalmente dalla tastiera) Output = emissione di dati all'esterno (generalmente a video)
	Elaborazione (generalmente Assegnazione)
	Linea di flusso
	Se • contiene una test • Ha una uscita nel caso in cui il test risulti vero •Ha una uscita nel caso in cui il test risulti falso •Ogni uscita può avere + istruzioni condizionate •Le 2 strade si ricongiungono sempre prima della istruzione successiva

Il primo PROBLEMA da risolvere

- Lo stesso problema può essere definito in modi diversi:

  Determinare la somma di 2 numeri immessi

  Calcolare la somma di 2 numeri dati

  Dati 2 numeri calcolarne la somma

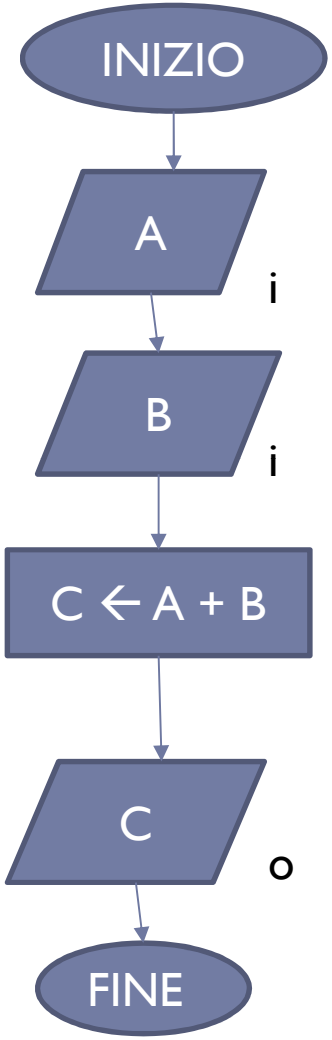
Somma di 2 numeri immessi

Viene immesso il primo numero in A

Viene immesso il secondo numero in B

Si determina la somma di A e B

Si stampa il risultato memorizzato in C



Struttura sequenziale

- Il diagramma precedente ha una struttura sequenziale

  cioè

  non ha costrutti condizionali né cicli iterativi.

- Tutti i blocchi saranno certamente eseguiti, dal primo all'ultimo

Le strutture condizionali

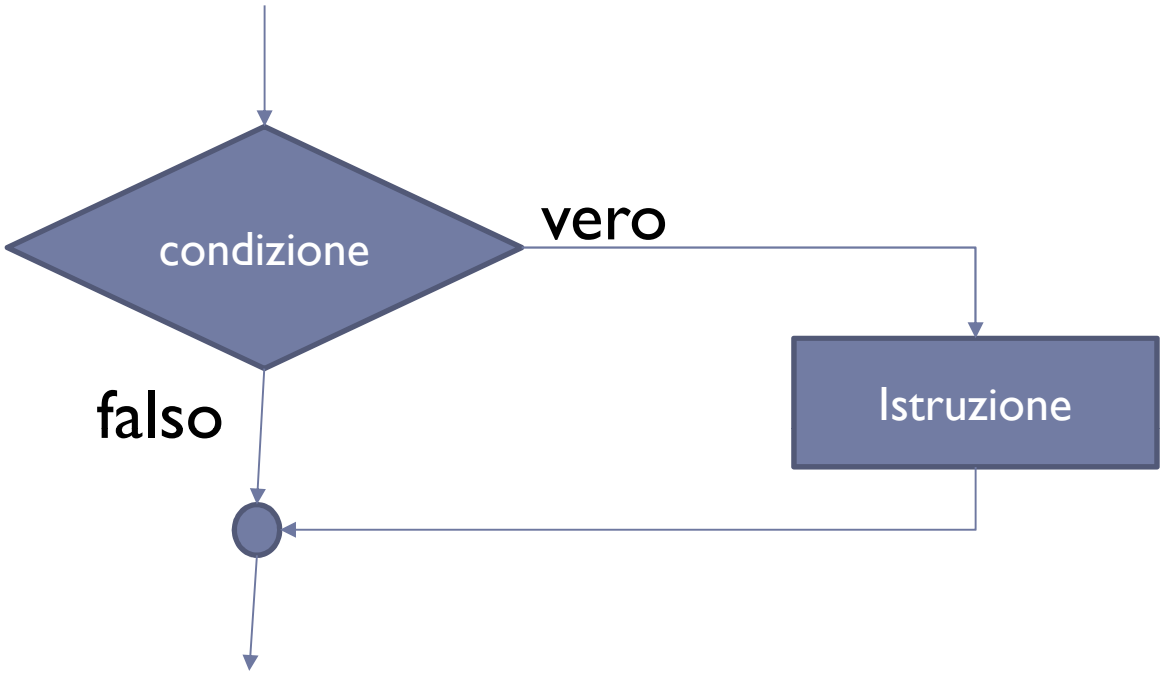
- Le strutture sequenziali non consentono di tracciare la logica risolutiva di qualunque algoritmo
- Spessissimo ci si trova davanti ad un bivio
- Giunti al quale bisogna scegliere se eseguire una istruzione o non eseguirla
- Viene testata una condizione. p.e. piove? Non piove? A=0? A<>0?
- Se il test risulta vero viene eseguita l'istruzione
- Se il test risulta falso l'istruzione condizionale non viene eseguita

```
se     piove
       apro l'ombrello
finese
…altre istruzioni
```

- FINESE indica la fine del costrutto SE
- Le istruzioni seguenti FINESE saranno eseguite in ogni caso

Condizioni

- A=0
- Nome='paolo'
- A<>B oppure A≠B
- X>Y
- Y<Z
- B>=C
- D<=E
- A not > B
- R not < S


condizione
vero
Istruzione
falso
altre istruzioni
eseguite comunque

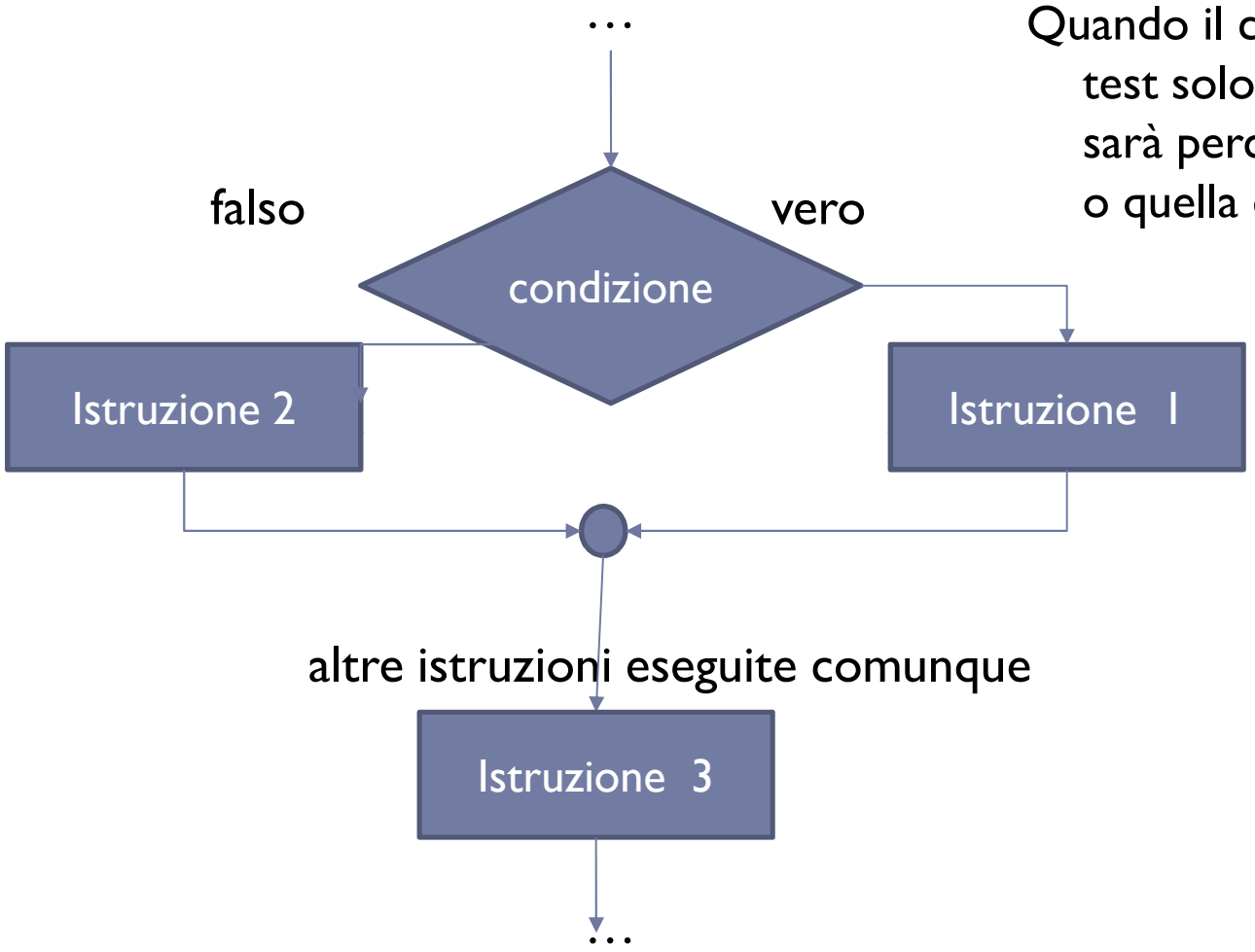
Le strutture alternative

- bisogna scegliere se eseguire una istruzione oppure eseguirne un'altra
- Viene testata una condizione. p.e. piove? Non piove? A=0? A<>0?
- Se il test risulta vero viene eseguita una istruzione
- Se il test risulta falso viene eseguita un'altra istruzione

```
se divisore = 0
        quoto = dividendo / divisore
altrimenti
        "divisione impossibile"
finese
…altre istruzioni
```

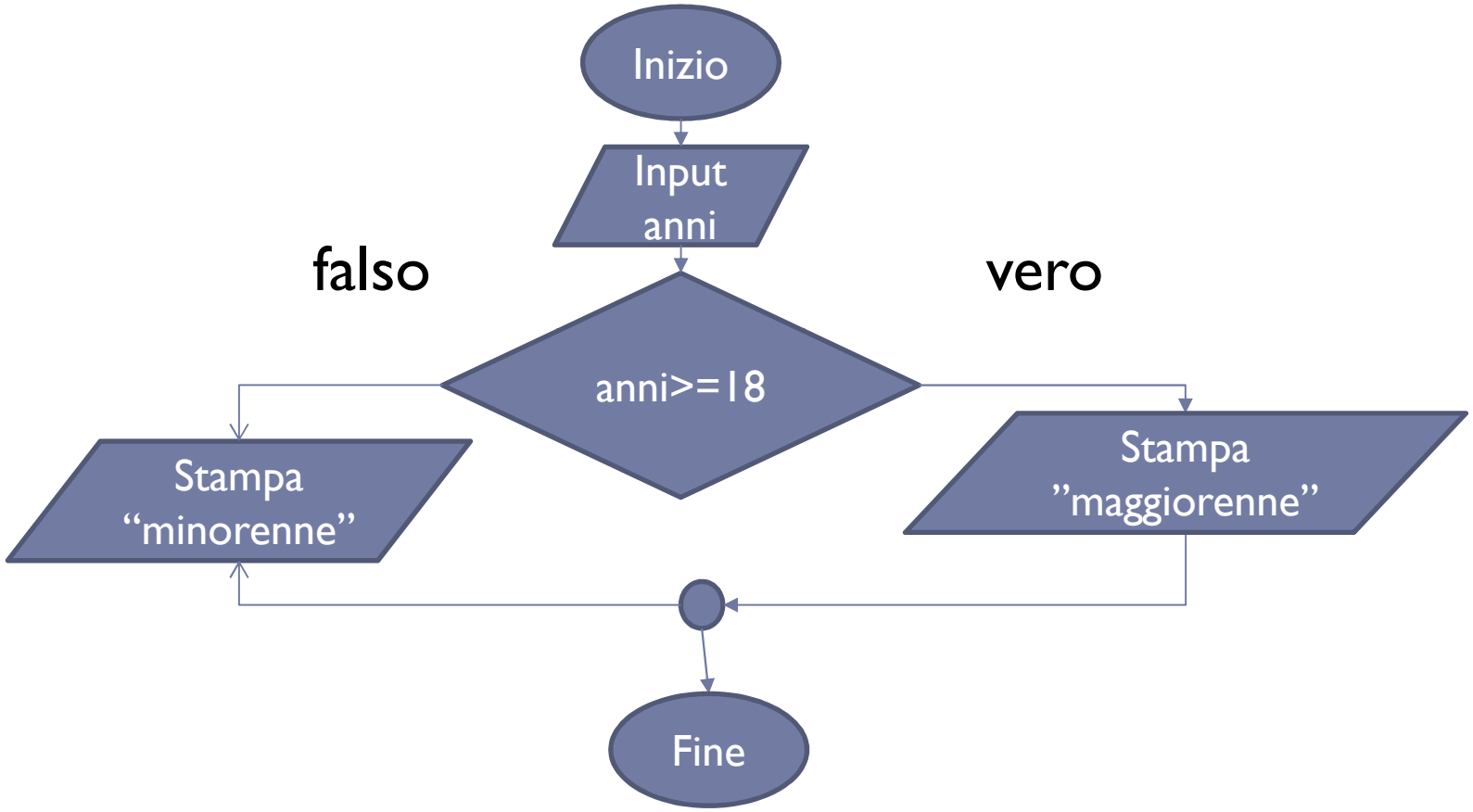
- FINESE indica la fine del costrutto SE
- Le istruzioni seguenti FINESE saranno eseguite in ogni caso

Esempio di struttura alternativa



Quando il controllo passerà dal test solo una delle 2 strade sarà percorsa: quella del vero o quella del falso

Esempio: Determinare se una età immessa è relativa ad un maggiorenne o minorenne

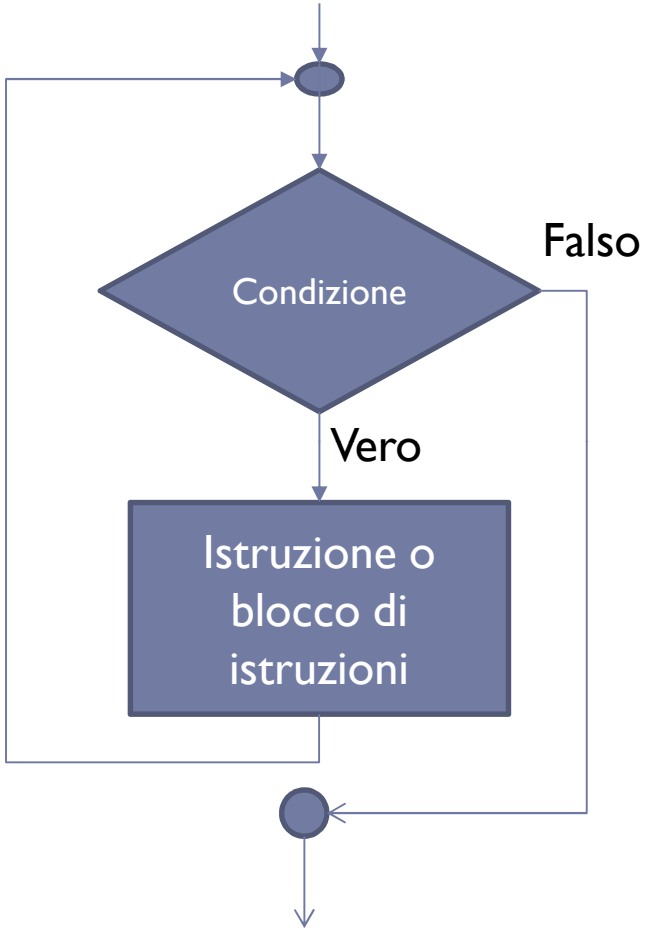


Strutture Iterative

- Le strutture iterative permettono di ripetere le istruzioni contenute al loro interno.
- I programmi diventano più brevi.
- Esempio:
  - Immettere 10 numeri diventa:
  - Ripeti 10 volte
    - Immetti un numero
  - Immettere una serie di numeri terminante con 0 diventa:
  - Inizio
    - Immetti un numero
  - Ripeti fintanto che il numero non è zero

WHILE – Controllo in testa

1)Verifica la condizione
2)Se vera
    entra nel ciclo
    esegue istruzioni
    ritorna a 1)
3)Se falsa
    esce dal ciclo
    esegue
    istruzioni successive



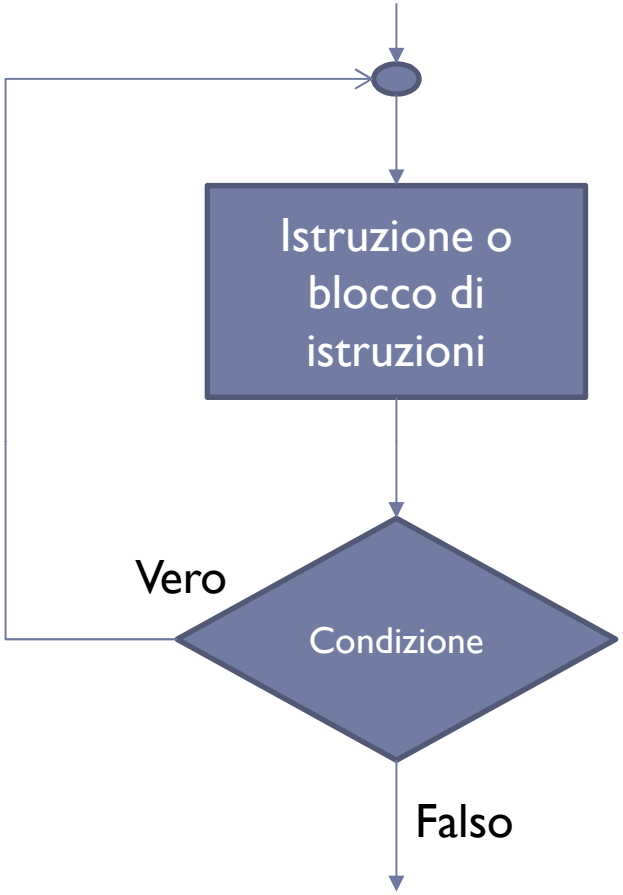
Ripeti…finché – Controllo in coda

1)entra nel ciclo
esegue istruzioni

2)Verifica la condizione
Se vera
ritorna a 1)

3)Se falsa
esce dal ciclo

4)esegue istruzioni successive



Link

- Code.org
  - Giochi da programmare
  - Labirinto
- Scratch
  - Ambiente grafico per iniziare a programmare
- techteach.altervista.com
  - I cicli in C/C++
  - Appunti di C++